

Программирование Python

Лекция 4

Кафедра ИВТ и ПМ

2017



План

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Прошлые темы

- ▶ Что такое вложенные циклы?
- ▶ Как они работают?



Прошлые темы

- ▶ Что такое вложенные циклы?
- ▶ Как они работают?
- ▶ Что такое **внешний цикл**?
- ▶ Что такое **вложенный (внутренний) цикл**?



Прошлые темы

- ▶ Что такое вложенные циклы?
- ▶ Как они работают?
- ▶ Что такое **внешний цикл**?
- ▶ Что такое **вложенный (внутренний) цикл**?
- ▶ Для чего нужен оператор `break`?
- ▶ Для чего нужен оператор `continue`?

Прошлые темы

- ▶ Как выйти сразу из всех циклов?

```
while ... and flag:
    while ... and flag:
        ...
    ...
...
...
```



Прошлые темы

- ▶ Как выйти сразу из всех циклов?

```
while ... and flag:
    while ... and flag:
        ...
    ...
...
...
```

Использовать оператор break и некоторый флаг

```
flag = True

while ... and flag:
    while ... and flag:
        ...
        if ... :
            flag = False
            break
    ...
```



Прошлые темы

- ▶ Что такое список (list)?
- ▶ Элементы какого типа он может хранить?
- ▶ Сколько элементов может содержать список?
- ▶ Как объявить пустой список?



Прошлые темы

- ▶ Что такое список (list)?
- ▶ Элементы какого типа он может хранить?
- ▶ Сколько элементов может содержать список?
- ▶ Как объявить пустой список?
`l = []`
- ▶ Как добавить элемент в конец списка?



Прошлые темы

- ▶ Что такое список (list)?
- ▶ Элементы какого типа он может хранить?
- ▶ Сколько элементов может содержать список?
- ▶ Как объявить пустой список?

```
l = []
```

- ▶ Как добавить элемент в конец списка?

```
l = l + [ 23 ]
```

или

```
l.append([ 23 ])
```

- ▶ Как узнать длину списка?



Прошлые темы

- ▶ Что такое список (list)?
- ▶ Элементы какого типа он может хранить?
- ▶ Сколько элементов может содержать список?
- ▶ Как объявить пустой список?

```
l = []
```

- ▶ Как добавить элемент в конец списка?

```
l = l + [ 23 ]
```

или

```
l.append([ 23 ])
```

- ▶ Как узнать длину списка?

```
len(l)
```



Прошлые темы

- ▶ Какой снippet будет работать быстрее?

```
l = [2, 4, 5, 7, 1, 41]
```

Вариант 1

```
for i in range(len(n)):  
    print( l[i] )
```

Вариант 2

```
for e in L:  
    print( e )
```

- ▶ Почему?



Прошлые темы

- ▶ Какой сниппет будет работать быстрее?

```
l = [2, 4, 5, 7, 1, 41]
```

Вариант 1

```
for i in range(len(n)):  
    print( l[i] )
```

Вариант 2

```
for e in L:  
    print( e )
```

- ▶ Почему?

Операция доступа к элементу списка по индексу(смещению) занимает время пропорциональное индексу



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Тернарный оператор

Тернарная (трёхместная) условная операция (от лат. ternarius — «тройной») - операция, возвращающая свой второй или третий операнд в зависимости от значения первого операнда.

Первый операнд обычно представлен логическим выражением.



Тернарный оператор

Общий вид тернарного оператора в Python:

`value1 if condition else value0`

Результатом будет `value1` если `condition == True`

Результатом будет `value0` если `condition == False`

Условием (`condition`) в тернарном операторе может быть любое логическое выражение



Тернарный оператор

```
res = "Yes" if flag else "No"
```

Использование этого оператора аналогично следующему коду

```
res = None
if flag:
    res = "Yes"
else:
    res = "No"
```



Тернарный оператор

Как будет работать следующий код?

```
res = ["No", "Yes"][flag]
```



Тернарный оператор

Как будет работать следующий код?

```
res = ["No", "Yes"][flag]
```

Первая пара квадратных скобок задаёт список.

Следующая пара - смещение(индекс) в этом списке.

Значение логической переменной `flag` будет преобразовано в целое число:

0 - если `flag == False`

1 - если `flag == True`

Таким образом из списка будет выбран элемент соответствующий значению `flag`.



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Динамическое создание списка

Чаще всего списки необходимо создавать автоматически.

Во-первых число элементов в списке может быть неизвестным и определяться в процессе наполнения списка.

Во-вторых между элементами списка и или их порядковым номером может быть некая зависимость, которую логично именно запрограммировать, а не описывать вручную каждый элемент.

Лучший способ создать список - использовать не цикл for или while, а **генератор списков**.

Генератор списков (list comprehension) - специальная конструкция языка для *создания* списков.



Генераторы списков

Синтаксис:

[выражение for перемен. in последовательность предикат]

Для каждого значения *переменной* из *последовательности* будет вычислено *выражение* и добавлено в список если предикат истинен.

Предикат можно опустить

Предикат (прогр.) - выражение, результат которого истина или ложь. За счёт внутренних оптимизаций генератор списков - это самый быстрый способ динамически создать список.



Генераторы списков

Создать список из квадратов чисел от 2 до 7

```
l = [ x**2 for x in range(2,8) ]
```

```
# l = [4, 9, 16, 25, 36, 49]
```

Для каждого значения x в последовательности `range(2,8)` будет вычислено выражение $x**2$, которое будет добавлено в список.

Синтаксически генераторы списков напоминают циклические конструкции, но тело цикла здесь находится перед заголовком цикла и всегда состоит из одного выражения

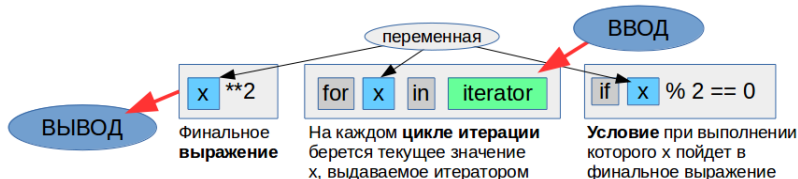


Генераторы списков

Создать список из квадратов только *чётных* чисел от 2 до 7

```
l = [x**2 for x in range(2,8) if x % 2 == 0]
```

```
# l = [4, 16, 36]
```



Генераторы списков

Для лучшей читаемости генератор списка стоит расположить на нескольких строках

```
l = [ x**2  
      for x in range(2,8)  
      if x % 2 == 0 ]
```

```
# l = [4, 16, 36]
```

Отступы в этом случае не создают вложенного блока.



Генераторы списков

Вложенные циклы

```
[ выражение  
  for Y in последовательность2  
  for X in последовательность1  
  предикат]
```

X, Y - переменные цикла

оператор `for X in последовательность1` является вложенным по отношению к `for Y in последовательность2`

Таким образом вложенные циклы в генераторах списков организуются таким же образом, что и обычные вложенные циклы



Генераторы списков

Задача: получить из вложенных списков (матрицы) - линейный список

```
matrix = [[0, 1, 2, 3],  
          [10, 11, 12, 13],  
          [20, 21, 22, 23]]
```

```
flattened = [n  
             for row in matrix  
             for n in row]
```

row - строка матрицы

```
print(flattened)  
# [0, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23]
```



Генераторы списков

Задача: создать матрицу

```
M = [ [ i*10+j for j in range(3) ] for i in range(3) ]
```



Генераторы списков

Задача: создать матрицу

```
M = [ [ i*10+j for j in range(3) ] for i in range(3) ]
```

Как будет выглядеть матрица?

```
[[0, 1, 2],  
 [10, 11, 12],  
 [20, 21, 22]]
```



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Кортежи

В Python **кортеж (tuple)** - неизменяемая упорядоченная коллекция объектов.

Кортежи во многом похожи на списки, кроме своей неизменности.

Создание пустого кортежа

```
t = ()
```

Кортеж из одного элемента

```
t = tuple(15)
```

```
t = (15,)
```

Запись *(15)* считается арифметическим выражением, поэтому так нельзя создать кортеж.



Кортежи

Кортеж из четырех элементов

```
T = (0, "Ni", 1.2, 3)
```

Еще один кортеж из четырех элементов

```
T = 0, "Ni", 1.2, 3
```

Кортеж можно создать из списка

```
t = tuple( [15, 23, 37, 48, 52] )
```

Кортеж можно создать из строки

```
t = tuple( "abc" )  
( 'a', 'b', 'c' )
```



Кортежи

Операции с кортежами

```
t = tuple( [15, 23, 37, 48, 52] )
```

- ▶ Доступ по индексу(смещению)

```
t[0]    # 15
```

```
t[-1]   # 52
```

- ▶ Доступ по срезу

```
t[0:2]  # (15, 23)
```

```
t[3:]   # (48, 52)
```

```
t[::2]  # (15, 37, 52)
```



Кортежи

Операции с кортежами

```
t = tuple( [15, 23, 37, 48, 52] )
```

- ▶ Конкатенация

```
t2 = t + (0, 1, e)
```

```
t3 = t3 + t
```

- ▶ Повторение

```
t = ('кря',) * 2  # ('кря', 'кря')
```



Кортежи

Операции с кортежами

```
t = tuple( [15, 23, 15, 15, 23] )
```

- ▶ Подсчёт числа вхождений

```
t2.count(15) # 3
```

```
t2.count(100) # 0
```

- ▶ Поиск элемента

```
t.index(значение)
```

```
t.index(значение, начало)
```

```
t.index(значение, начало, конец)
```

Если значения нет в кортеже, то возникает ошибка времени выполнения

```
ValueError: tuple.index(x): x not in tuple
```



Кортежи

Нельзя изменять *содержимое* отдельных элементов в кортеже.
Только если это не словарь, множество или список.

`T = (1, [2, 3], 4)`

`T[1] = [9,9]` *# Ошибка!*

`T[1][0] = 0` *# Так можно*



Кортежи

Операции с кортежами

```
t = tuple( [15, 23, 15, 15, 23] )
```

- ▶ проверка вхождения элемента

```
23 in t    # True
```

```
99 in t    # False
```



Кортежи

Обход всех значений в кортеже происходит также как и в списке

```
T = tuple( [15, 23, 15, 15, 23] )
```

```
for e in T:  
    print( e )
```

Кортеж нельзя менять, поэтому доступ по индексу в цикле используется редко.



Кортежи

Когда использовать кортежи

- ▶ Когда нужен неизменяемый объект
Например когда нужен неизменяемый список.
- ▶ Когда нужно представить набор разнородных значений
Кортежи можно использовать вместо структурированных типов (record в Pascal или struct в C++)

Например данные о погоде можно хранить в таком кортеже
('Чита' , (113.466667, 52.05), (5,-6), 'Ясно')

Списки рекомендуется использовать для хранения набора значений одинакового типа, в то время как кортежи для хранения значений разных типов, в том числе других кортежей.



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Множественное (групповое) присваивание

Чтобы объявить (создать) переменную в Python нужно обязательно задать ей начальное значение:

```
a = 10
```

```
b = 10
```



Множественное (групповое) присваивание

Чтобы объявить (создать) переменную в Python нужно обязательно задать ей начальное значение:

```
a = 10  
b = 10
```

Если значения переменных одинаковы, то можно использовать **множественное присваивание**:

```
a = b = 10
```

При этом переменные *a* и *b* будут совершенно независимы.



Кортежное присваивание

Кортеж - упорядоченный набор фиксированной длины

Кортежное присваивание:

$$a, b = 7, 13$$

Такая операция присваивания аналогична следующей:

$$a = 7$$
$$b = 13$$

В операции кортежного присваивания число переменных слева должно соответствовать числу выражений справа. (Если не используется распаковка)



Кортежное присваивание

В правой части может быть выражение.

```
a, b = 10.5, 2+2
```

```
a, b = sin(pi), 10000
```

```
a, b = 'Hello', 42
```

Как и в обычной операции присваивания, правая часть вычисляется перед тем как будет записана в переменную. В

левой части должны быть только переменные, не выражения

```
a+1, b = 7, 45    # Ошибка!
```

```
a+b, c = 20, 99   # Ошибка!
```



Ещё один способ обмена значений переменных

$a, b = b, a$

Как это работает:

1. Вычисляется правая часть (после знака равно)
2. В памяти создаётся новый кортеж (назовём его a' , b') из значений b и a
3. Поэлементно выполняется операция присваивания

$a = a'$

$b = b'$



Присваивание последовательностей

```
a,b = [42,43]
```

```
# a = 42
```

```
# b = 43
```

```
a,b,c = 'qwe'
```

```
# a = 'q'
```

```
# b = 'w'
```

```
# c = 'e'
```



Комбинированные инструкции присваивания

$X = 10$

$Y = 3$

Как увеличить X на Y ?



Комбинированные инструкции присваивания

$X = 10$

$Y = 3$

Как увеличить X на Y ?

$X = X + Y$

Как увеличить X в Y раз?



Комбинированные инструкции присваивания

$X = 10$

$Y = 3$

Как увеличить X на Y ?

$X = X + Y$

Как увеличитать X в Y раз?

$X = X + Y$

Эти, а также аналогичные операции, в Python можно записывать короче:

$X += Y$ $\# X = X + Y;$ $X = 13$

$X *= Y$ $\# X = X * Y;$ $X = 30$



Комбинированные инструкции присваивания

Другие примеры

$X -= Y$ *# $X = X - Y$; $X = 7$*
 $X /= Y$ *# $X = X / Y$; $X = 3.33333$*
 $X //= Y$ *# $X = X // Y$; $X = 3$*
 $X \%= Y$ *# $X = X \% Y$ $X = 1$*
 $X **= Y$ *# $X = X ** Y$ Возведение X в степень Y ;*

Побитовые операции

$X \&= Y$ *# И*
 $X |= Y$ *# ИЛИ*
 $X \wedge= Y$ *# XOR (исключающее ИЛИ)*
 $X >>= Y$ *# побитовый сдвиг вправо*
 $X <<= Y$ *# побитовый сдвиг влево*



Операции присваивания

Кортежное присваивание

```
x,y = 7,20
```

Обмен значений переменных

```
x,y = y,x
```

```
a,b = [42,43]
```

```
a,b,c = 'qwe'
```

```
a,b,*c = 1,2,3,4,5 # c = [3, 4, 5]
```

```
*a,b,c = 1,2,3,4,5 # a = [1, 2, 3]
```

```
a,*b,c = 1,2,3,4,5 # b = [2, 3, 4]
```



Кортежи и списки

Как представить набор координат точек на плоскости?



Кортежи и списки

Как представить набор координат точек на плоскости?

```
Points = [ (1,3), (0,4), (2,1), (-1,1) ]
```

```
for x,y in Points:  
    print( sqrt(x**2 + y**2) )
```

```
3.1622776601683795
```

```
4.0
```

```
2.23606797749979
```

```
1.4142135623730951
```



Кортежи и списки

Как организовать цикл изменяющий координаты точек?



Кортежи и списки

Как организовать цикл изменяющий координаты точек?

```
Points = [ (1,3), (0,4), (2,1), (-1,1) ]
```

```
for i in len(Points):  
    x,y = Points[i]  
    Points[i] = y,x
```

Не смотря на то, что кортеж это неизменяемая коллекция, заменять один кортеж другим можно.



Outline

Прошлые темы

Операторы

Тернарный оператор

Списки

Генераторы списков

Кортежи

Операция присваивания

Сообщения об ошибках



Сообщения о синтаксических ошибках

На какие две категории можно разделить ошибки в программах?

- ▶ Ошибки выявляемые компилятором или интерпретатором
- ▶ Логические ошибки
Эти ошибки должен выявлять программист



NameError

Что общего у всех этих операторов?

```
print( x )  
b = a * 2  
c = sin( b )  
z = randint(0,99)  
sleep(10)  #приостанавливает программу на 10 секунд
```

Предполагается, что в этом сниппете никакие другие операторы не пропущены.



NameError

Что общего у всех этих операторов?

```
print( x )  
b = a * 2  
c = sin( b )  
z = randint(0,99)  
sleep(10)  #приостанавливает программу на 10 секунд
```

Предполагается, что в этом сниппете никакие другие операторы не пропущены.

NameError: name 'x' is not defined.

NameError: name 'a' is not defined.

NameError: name 'sin' is not defined.

NameError: name 'randint' is not defined.

NameError: name 'sleep' is not defined.

NameError - обращение к не объявленным именам
(идентификатор)



SyntaxError - общие ошибки синтаксиса

```
name = "Петя"  
print name
```



SyntaxError - общие ошибки синтаксиса

```
name = "Петя"  
print name
```

SyntaxError: invalid syntax

Параметры функции должны быть в круглых скобках.

Правильно:

```
print( name )
```

Примечание: синтаксис второй версии Python допускал такой способ вывода переменной на экран.



SyntaxError - общие ошибки синтаксиса

```
from random import randint
a = randint(0,9)
if a == 5
    print("a равно пяти")
```

SyntaxError: invalid syntax



SyntaxError - общие ошибки синтаксиса

```
from random import randint
a = randint(0,9)
if a == 5
    print("a равно пяти")
```

SyntaxError: invalid syntax

Перед каждым блоком в составном операторе должно стоять двоеточие

Правильно:

```
if a == 5 :
    print("a равно пяти")
```



SyntaxError - общие ошибки синтаксиса

```
from random import randint  
a = randint(0,9)  
if a = 5 :  
    print("а равно пяти")
```



SyntaxError - общие ошибки синтаксиса

```
from random import randint
a = randint(0,9)
if a = 5 :
    print("a равно пяти")
```

SyntaxError: invalid syntax

В условном операторе должно быть логическое выражение.
Здесь - вместо оператора проверки на равенство использован оператор сравнения.

Правильно:

```
if a == 5 :
    print("a равно пяти")
```



IndentationError - ошибки связанные с отступами

```
a = 10  
if a > 0:  
print(a)
```



IndentationError - ошибки связанные с отступами

```
a = 10  
if a > 0:  
print(a)
```

IndentationError: expected an indented block.

```
a = 10  
if a > 0:  
    print(a)
```



TabError - ошибки связанные с отступами

```
a = 10  
    print(a)
```

TabError: inconsistent use of tabs and spaces in indentation

Правильно

```
a = 10  
print(a)
```



TypeError - ошибки типа

```
a = input("a> ")  
b = a ** 3
```



TypeError - ошибки типа

```
a = input("a> ")  
b = a ** 3
```

TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int' Правильно:

```
a = float( input("a> ") )  
b = a ** 3
```



Ссылки и литература

- ▶ Всё о выражениях-генераторах, генераторах списков, множествах и словарей
- ▶ StackOverflow: What's the difference between lists and tuples?



Ссылки и литература

ЭБС

- ▶ biblio-online.ru - ЭБС Юрайт
- ▶ studentlibrary.ru - ЭБС "КОНСУЛЬТАНТ СТУДЕНТА"
- ▶ Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата Содержит краткое описание языка.
- ▶ ru.wikibooks.org/wiki/Python - Викиучебник
- ▶ Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- ▶ Официальная документация Python3
`help(имя)`



Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming

